UserMap – an Adaptive Enhancing of User-Driven XML-to-Relational Mapping Strategies

Irena Mlynkova, Jaroslav Pokorny

{irena.mlynkova,jaroslav.pokorny}@mff.cuni.cz

Charles University

Faculty of Mathematics and Physics

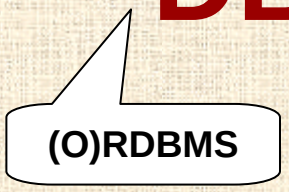
Department of Software Engineering

Prague, Czech Republic

Introduction

- XML = a standard for data representation and manipulation
  - ⇒ A boom of implementations
    - XML file systems, native XML databases, XML-enabled databases, …
- XML-enabled databases – most practically used
  - Less efficient than native XML databases
  - Exploitation of tools and functions of traditional (O)RDBMS
    - Reliable and robust
    - Long theoretical and practical history
  - Major DB vendors support XML, SQL standard: new part SQL/XML

DB-Based XML Processing Methods (1)



- Key concern: Choice of the most efficient XML-to-relational mapping strategy
- Various classifications:
  - Generic (schema-oblivious) vs. schema-driven – omitting vs. exploiting XML schema
  - Fixed vs. adaptive – mapping on the basis of data model vs. target application
  - User-defined vs. user-driven – the amount of user involvement
    - User specifies target schema and required mapping vs. user locally modifies a default mapping

$\Rightarrow$ Which approach is the best?

DB-Based XML Processing Methods (2)

- Problem: No universally efficient approach
  - Various applications: Updatability of data, redundancy, special XML data formats (RDF, XHTML, …)
    - Requirements which collide
    - Efficiency vs. space overhead
- The most promising approaches: adaptive and user-driven methods
  - Adapt the mapping to a target application
    - Sample XML documents + sample XML queries
    - Schema annotations with required mapping
    - …

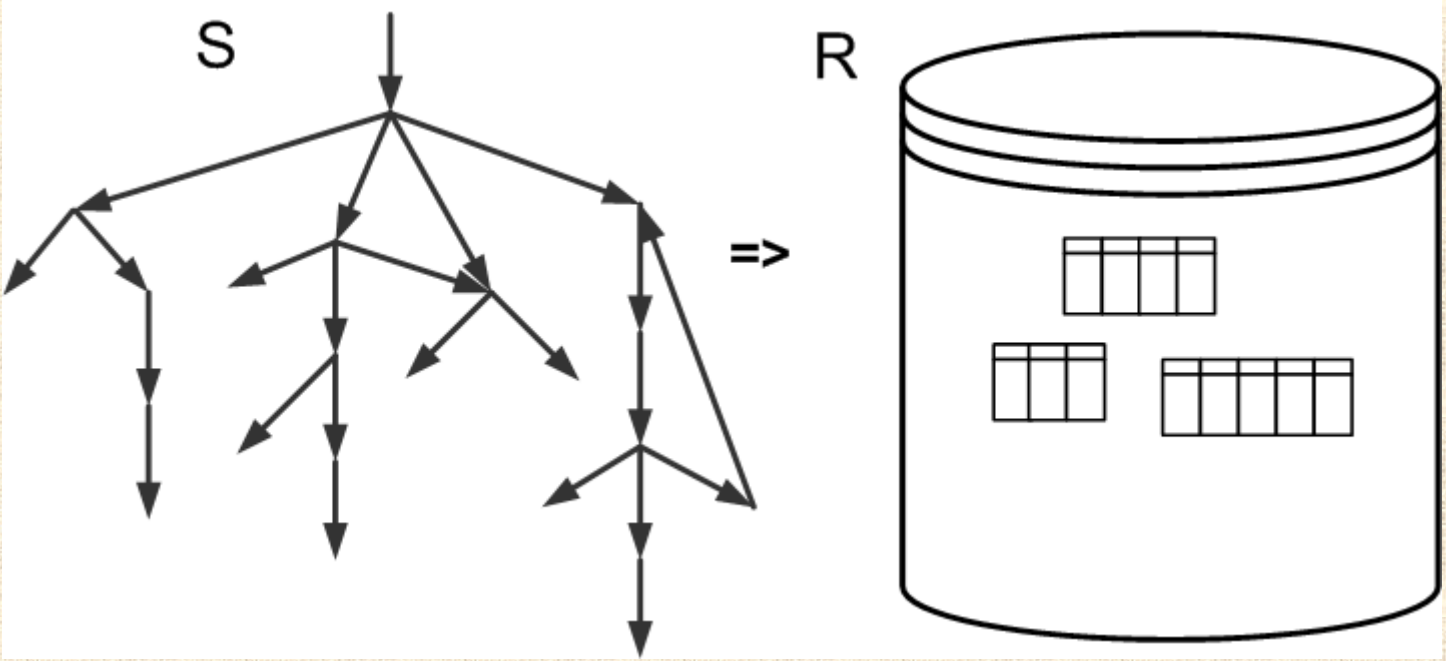
⇒ Is it possible to improve them?

Mlynkova, Pokorny: Adaptability of Methods for Processing XML Data using Relational Databases – the State of the Art and Open Problems. IJCSA, volume 4, number 2, 2007. ISSN 0972-9038.

UserMap

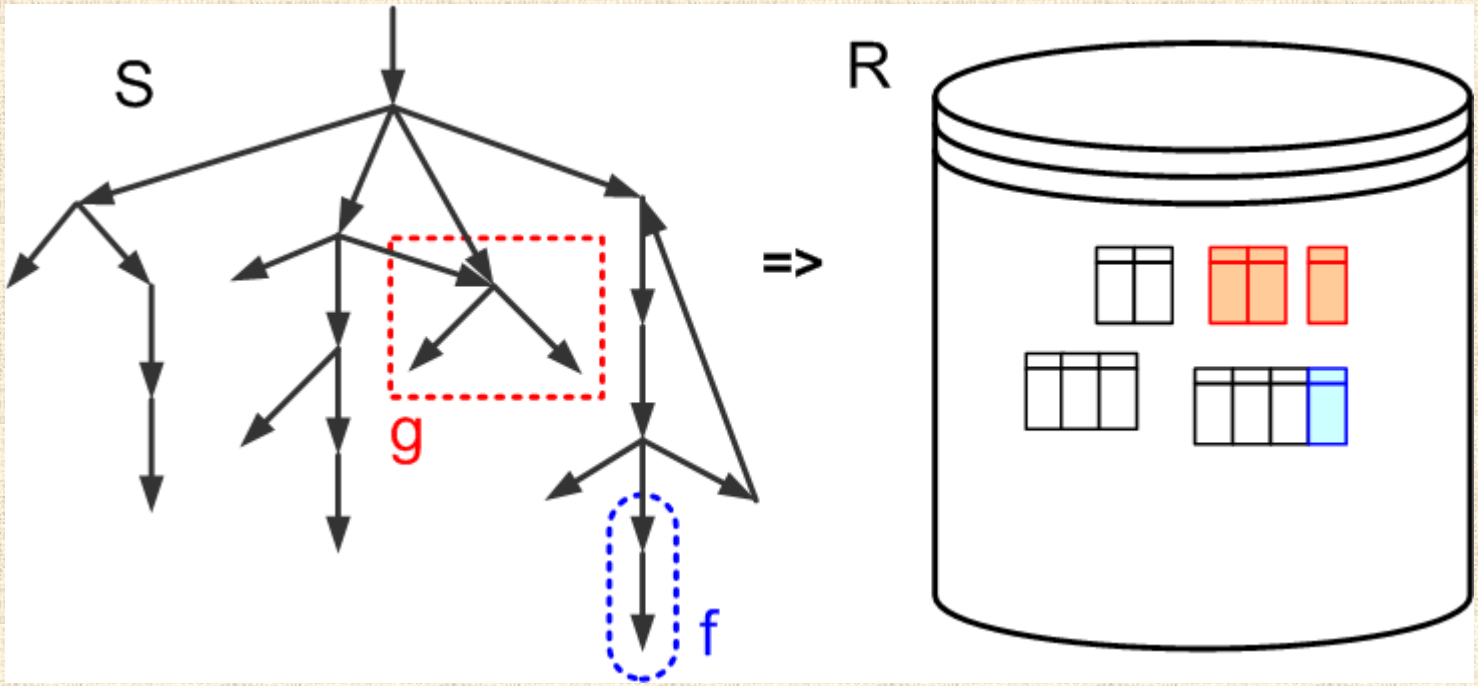
- Experimental implementation
- Improvements of user-driven methods
  - Several related problems
    - Similarity of XML data, adaptive strategy, query evaluation, user interaction, derivation of XML schema, …
- Goals of this presentation:
  - Basic ideas and solved problems
  - Current issues
  - Open problems

Fixed Schema-Driven Mapping



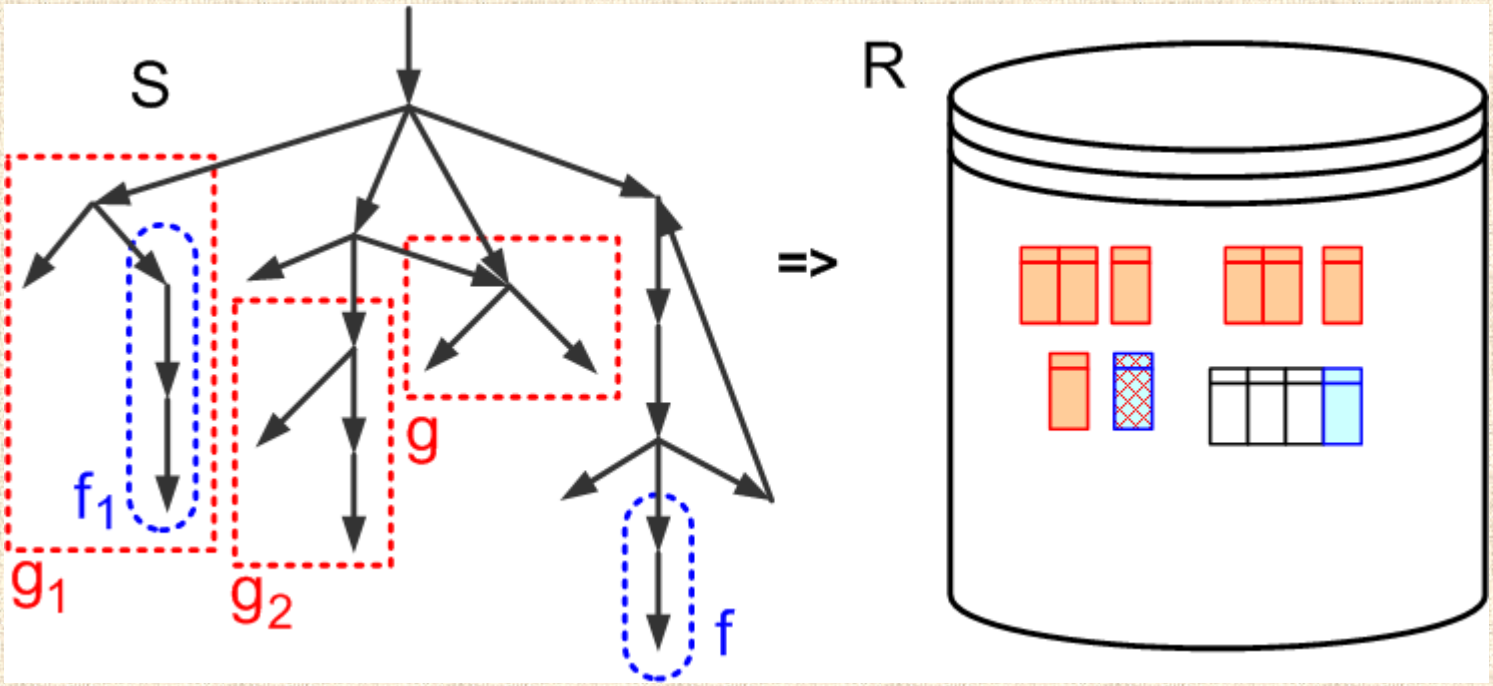
- **Aim: No redundancy, no null values, no dependencies, … ⇒ 4NF**
- **Note: Generic methods view XML documents as trees = a kind of schema as well**

User-Driven Mapping



- **A default fixed mapping strategy**
- **User annotates subschemes with required mapping modifications**

UserMap Mapping



- **UserMap can find new schema annotations = help the mapping process**

Basic Observations

- **Weak exploitation of user-given information – annotations are just directly applied**
  - **Idea: Annotations = "hints" how to store particular XML patterns ⇒ we can store similar data in a similar way**
- **Default mapping strategy is always fixed**
  - **Idea: Adaptive strategy**

⇒ **Emphasis on user-given information and similarity**

Solved Problems

- Adaptive strategy based on similarity of XML data
  - No sample XML documents, XML queries

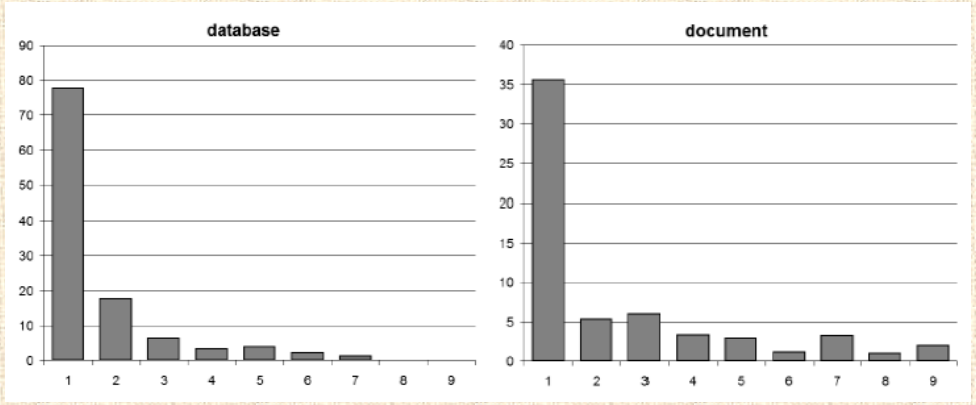
Mlynkova: A Journey towards More Efficient Processing of XML Data in (O)RDBMS. CIT 2007, Aizu-Wakamatsu, Japan. IEEE CS Press, 2007. ISBN 0-7695-2983-6.

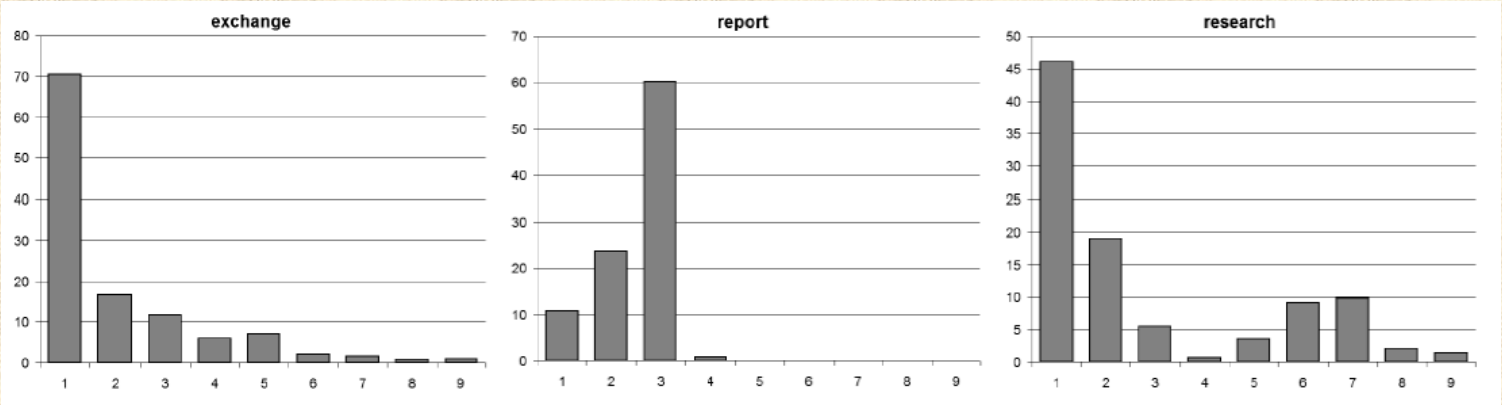
- Similarity function and search algorithm
  - Schema-level similarity, algorithm for tuning weights, search heuristics

Mlynkova, Pokorny: Similarity of XML Schema Fragments Based on XML Data Statistics. IIT 2007, Dubai, United Arab Emirates. IEEE CS Press, 2007. ISBN 978-1-4244-1841-1.

- Advantages:
  - User is not forced to annotate all schema fragments
  - System can reveal new structural similarities

Results





- **Iterative search for similar schema fragments**
  - **5 categories of real-world XML schemes**
    - **% of annotated fragments in each iteration**
  - **4 iterations on average**
  - **% of annotated fragments depends on category and data**

Mlynkova, Toman, Pokorny: Statistical Analysis of Real XML Data Collections. COMAD 2006, New Delhi, India. Tata McGraw-Hill Publishing Co. Ltd., 2006. ISBN 0-07-063374-6.

Current Issues

1. **Correction of the set of candidates for annotations**
   - Meaningless, multiple choices, …
   - Interaction with a user
2. **Query evaluation**
   - Interface between various storage strategies within a single schema, how to deal with redundancy

- **Both issues are related to user-driven methods in general**
  - Existing systems support only simple mapping strategies $\Rightarrow$ the solutions are trivial
- **To demonstrate the problems: Sample set of annotations**
  - Represent several types of mapping strategies

Attribute	Value	Function
INOUT	inline, outline	The fragment is inlined or outlined to/from parent table.
GENERIC	edge, attribute, universal	The fragment is stored using schema-oblivious Edge, Attribute, or Universal strategy (Florescu & Kossmann 1999).
SCHEMA	basic, shared, hybrid	The fragment is stored using schema-driven Basic, Shared, or Hybrid strategy (Shanmugasundaram et al. 1999).
TOCLOB	true	The fragment is stored to a CLOB column.
INTERVAL	true	The fragment is indexed using the Interval encoding (Yoshikawa et al. 2001).

Annotations

- Problem: Annotated fragments do intersect
  - Not all intersections are meaningful
  - Some intersections cause multiple choices
- General types of intersections:
  - Redundant – both methods are applied
    - XHTML fragments ⇒ shredding into tables + CLOB
  - Overriding – only one of the methods is applied
    - Classical user-driven strategies – local mapping changes
  - Influencing – both methods are combined into a single one
    - Shredding + indices/numbering schemes
- The system must know allowed types intersections
  - For all particular subsets of annotations
  - For all particular orders of intersection

Overriding and redundant intersections

	INOUT	GENERIC	SCHEMA	TOCLOB	INTERVAL
INOUT	∅	×	×	×	×
GENERIC	×	∅	✓	×	×
SCHEMA	×	✓	∅	×	×
TOCLOB	×	✓	✓	∅	×
INTERVAL	×	×	×	×	∅

Influencing intersection

	INOUT	GENERIC	SCHEMA	TOCLOB	INTERVAL
INOUT	∅	✓	✓	×	×
GENERIC	×	∅	×	×	×
SCHEMA	×	×	∅	×	×
TOCLOB	×	×	×	∅	×
INTERVAL	×	✓	✓	×	∅

Examples:

- **INOUT can be applied only on a king of shredding**
- **TOCLOB can be applied only on a mapping strategy, not vice versa**

The number of allowed options is low

```
<xs:element name="Actor"
            usermap:SCHEMA="hybrid">
 <xs:complexType>
  <xs:sequence>

   <xs:element name="Name"
               usermap:TOCLOB="true">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="FirstName"
                  type="xs:string"/>
      <xs:element name="LastName"
                  type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>

   ...
```

```
<xs:element name="Filmography">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Movie"
               maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="Title"
                  type="xs:string"
                  usermap:INOUT="outline"/>
      <xs:element name="Year"
                  type="xs:int"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>

 </xs:sequence>
</xs:complexType>
</xs:element>
```

Example…

…example

Actor(ID:integer,
FirstName:string,
LastName:string,
parentID:integer)
Movie(ID:integer,
Title:string,
Year:int,
parentID:integer)

Pure Hybrid mapping

Actor(ID:integer,
Name:CLOB,
parentID:integer)
Movie(ID:integer,
Year:int,
parentID:integer)
Title(ID:integer,
Title:string,
parentID:integer)

TOCLOB overrides SCHEMA

Multiple options ⇒ user interaction

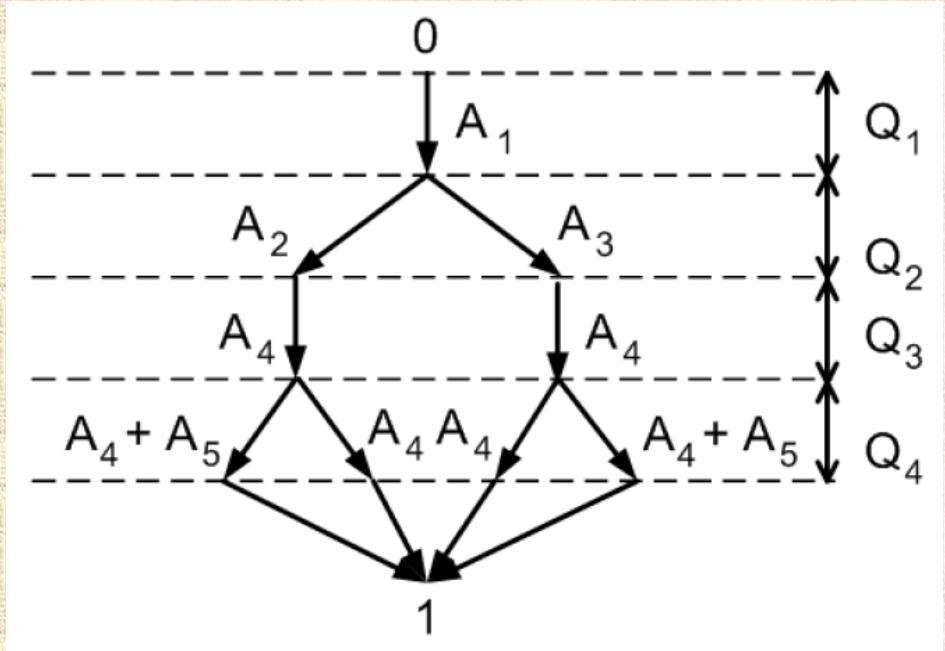
Actor(ID:integer,
Name:CLOB,
FirstName:string,
LastName:string,
parentID:integer)
Movie(ID:integer,
Year:int,
parentID:integer)
Title(ID:integer,
Title:string,
parentID:integer)

TOCLOB is redundant

Query Evaluation (1)

- **Problems:**
  - **Interface between two mapping strategies**
  - **Redundant and influencing intersections ⇒ multiple ways of evaluation**
- **Idea: Structural tables – carry information about mapping**
  - **For each element and attribute we know where it is stored (tables, columns) + related details (data types, indices)**
- **Two types of annotations:**
  - **Early binding – processed before schema is mapped**
    - **Changes of structure of the target schema**
    - **e.g. INOUT, TOCLOB**
  - **Late binding – processed as late as a query is evaluated**
    - **Additional information**
    - **e.g. INTERVAL**

Query Evaluation (2)



- **Late-binding and redundant intersections ⇒ multiple query plans**
- **Evaluation graph:**
  - **Contains all possible paths of evaluation of query Q**
    - **Divided into parts Q_1, Q_2, …, Q_k according to annotations A_1, A_2, …, A_l and/or their combinations, i.e. mapping strategies**
  - **Edge = possible evaluation strategy**
  - **Node = interface between two strategies**
  - **Length of edge = cost of evaluation using respective strategy + cost of interface between two strategies**
- **We search for the shortest path**
- **Simple, general solution ⇒ individual optimizations?**

Conclusions and Future Work

- Simple idea of user-driven methods and their improvement ⇒ plenty of related problems to be solved
- UserMap supports sample representatives to demonstrate the problems and possible solutions
- Future work:
  - Generalization and optimization of the ideas
    - Especially query evaluation
  - Combination with classical adaptive methods
    - Too many input information (documents, queries, annotations), but better results
  - Simplification of inputs
    - User can specify more general aspects (e.g. exploitation updates) than exact mapping strategies or precise queries
  - Dynamic adaptability
    - The required information can be gathered on the fly
    - The schema can be adapted according to changing application

Thank you